Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего образования

**«Пермский национальный исследовательский**

**политехнический университет»**

Кафедра «Информационные технологии и автоматизированные системы»

**О Т Ч Ё Т**

**по лабораторной работе №19**

Дисциплина: Основы теории алгоритмов и структуры данных.

Тема: “ Организация стека с помощью класса ”

Вариант.12

Выполнила работу:

студент группы ИВТ-20-2Б

Ананина Арина Юрьевна

Проверила:

Доцент кафедры ИТАС

Полякова О.А.

Пермь, 2021

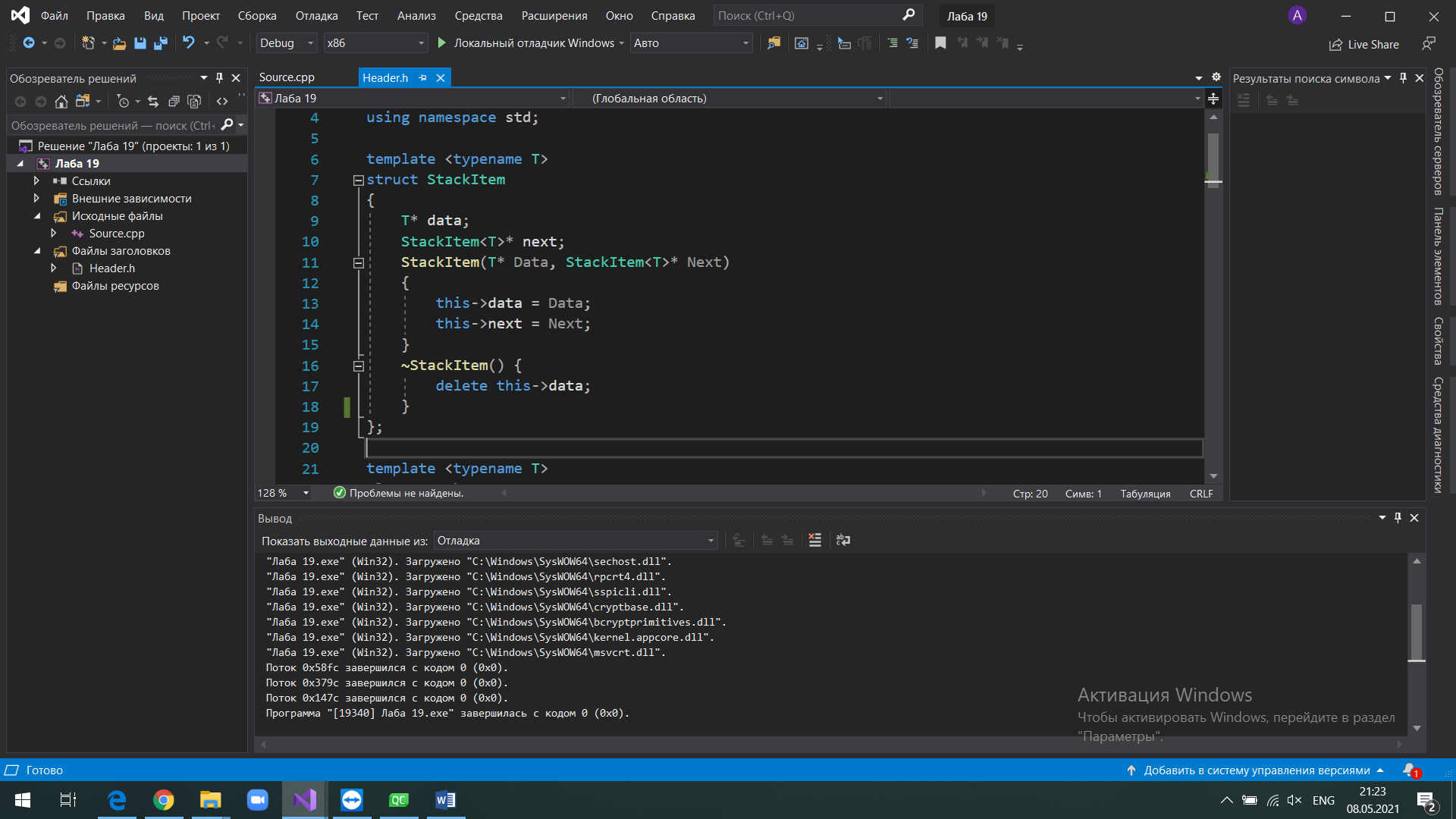
1. **Постановка задачи**

Организовать абстрактный тип данных – стек с использованием классов.

1. **Анализ работы**
2. Какие типы данных использовались в ходе работы.
   1. Использовался тип данных int, переменные этого типа использовались в качестве данных, хранимых в стеке, и для хранения размера стека.
3. В каком виде данные будут представлены для решения задачи.
   1. Данные будут представлены в виде структуры и класса.

В структуре StackItem будут храниться данные элементов стека, а в классе Stack будет храниться ссылка на первый элемент стека – ссылка на элемент структуры.

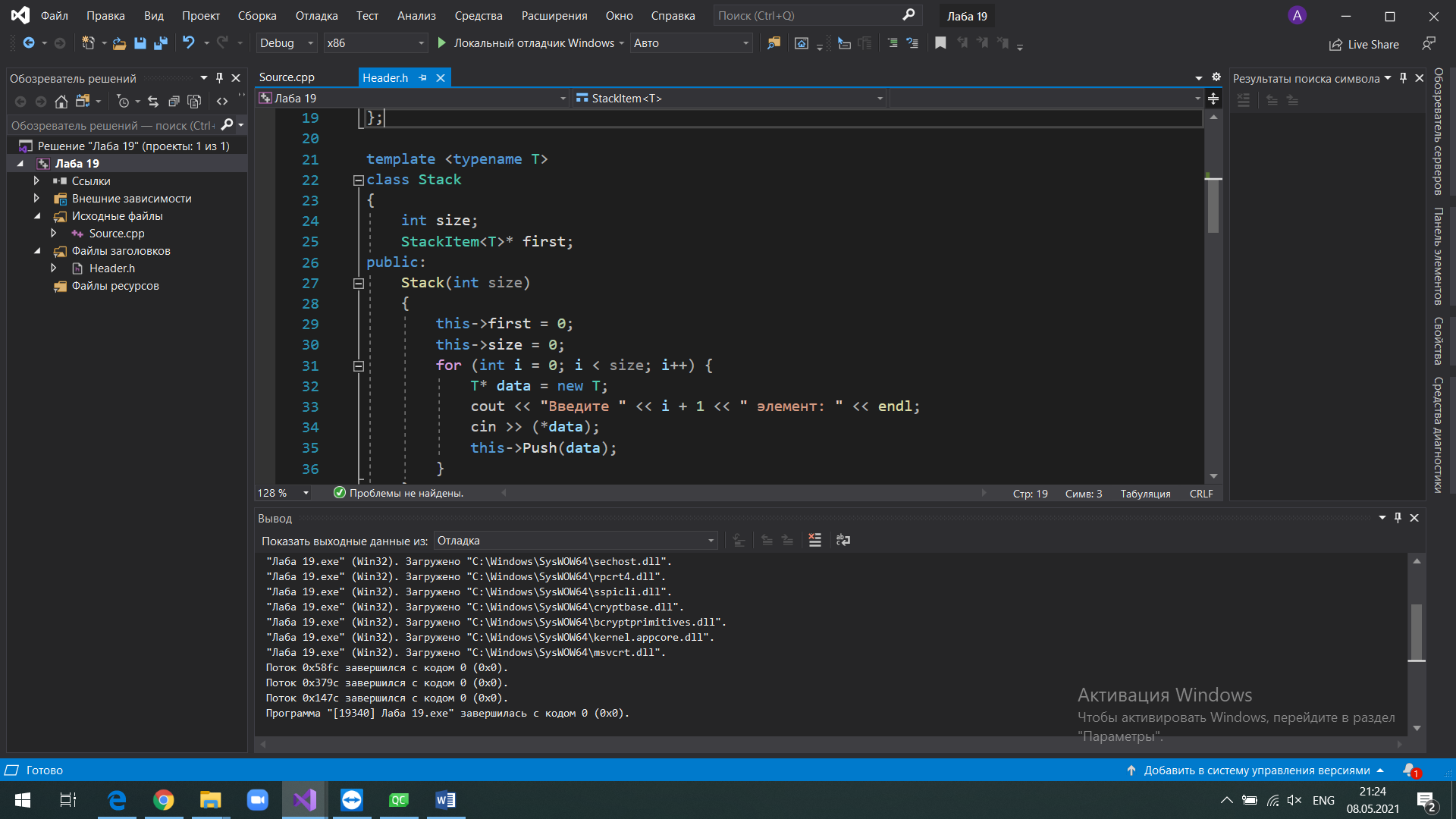
1. Поля, используемые в классе и структуре.
   1. Поля структуры StackItem:



Поле data хранит информацию элемента.

Поле next хранит ссылку на следующий элемент стека.

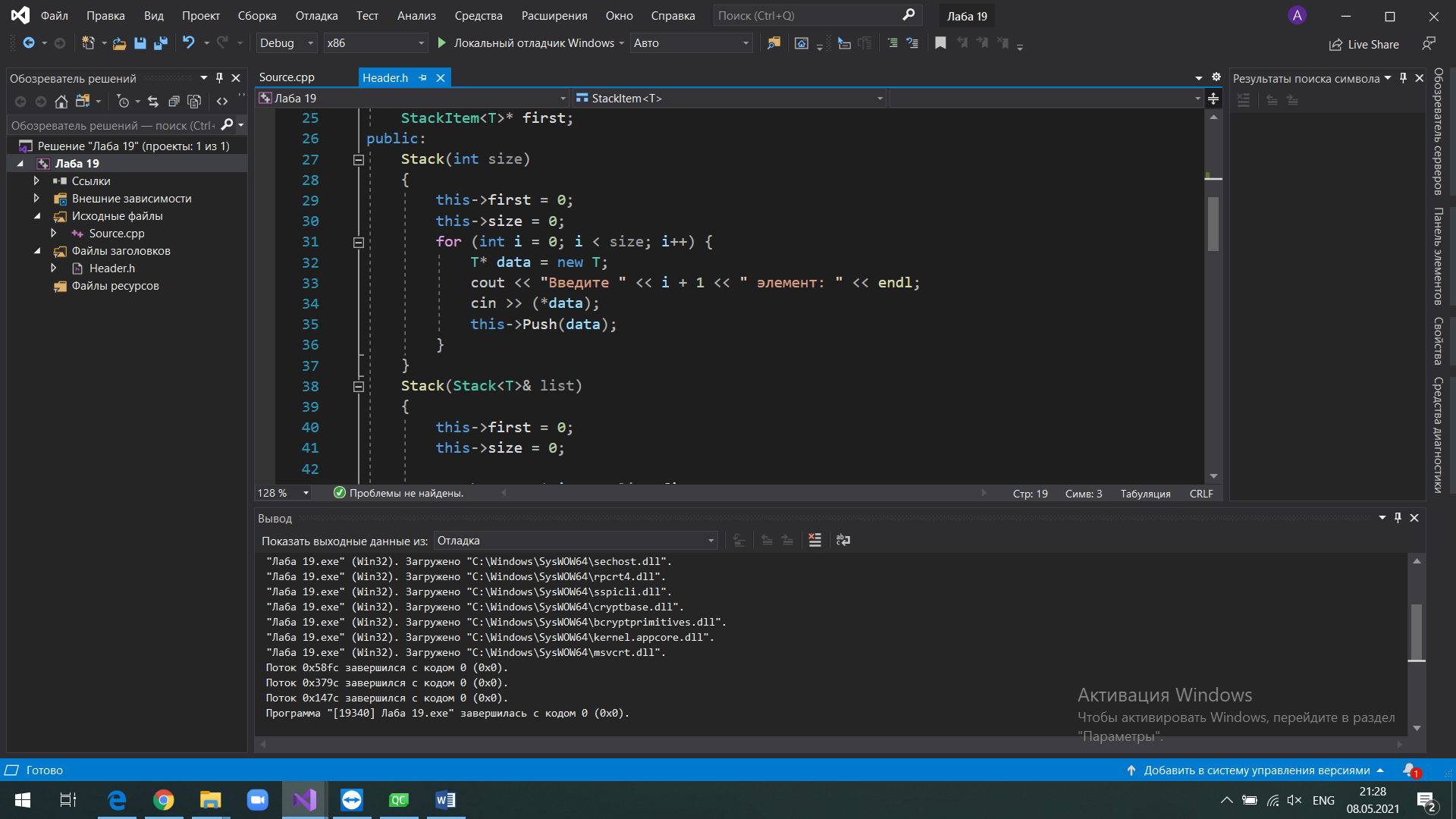
* 1. Поля класса Stack:

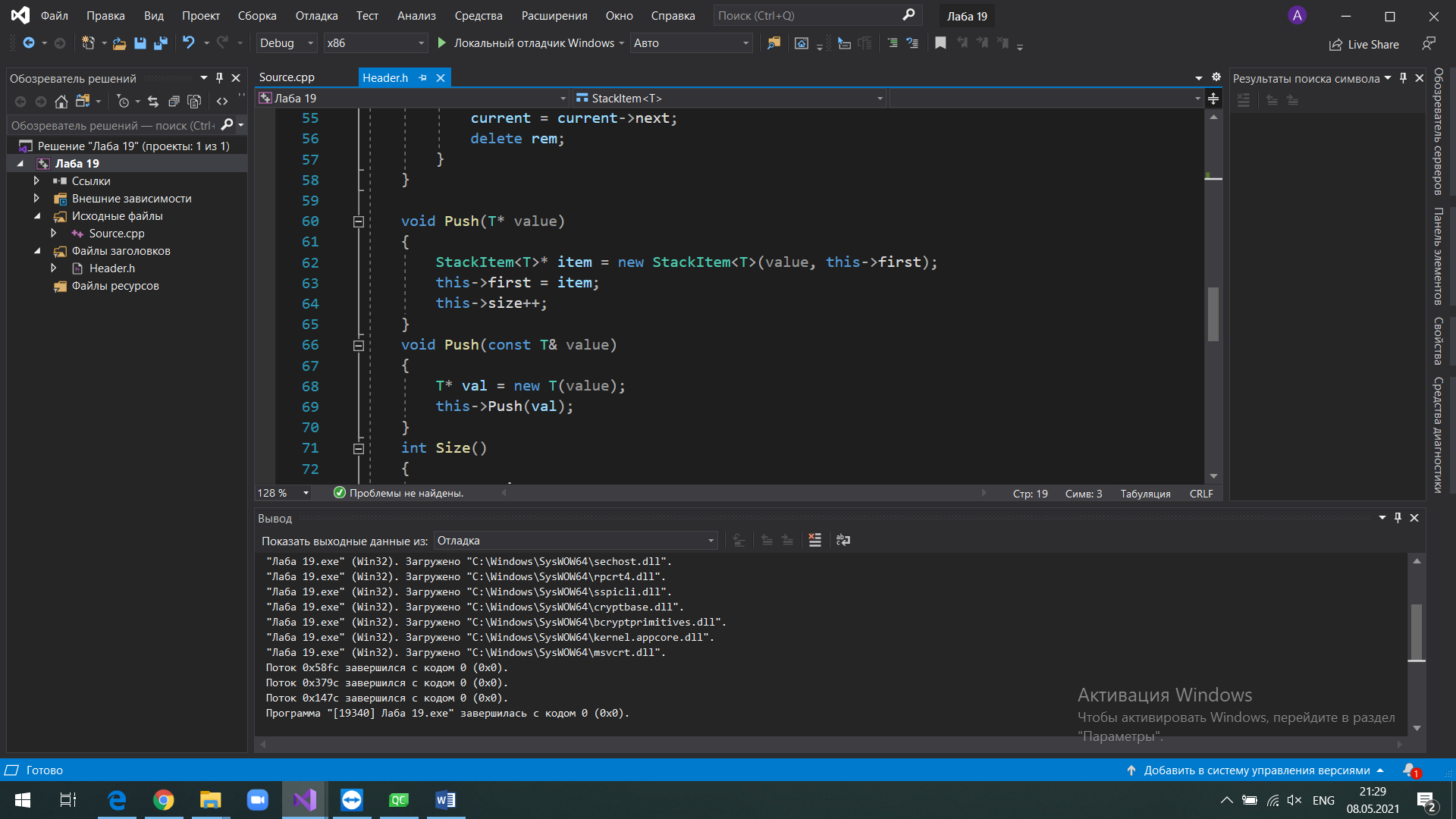


Поле size хранит размер стека.

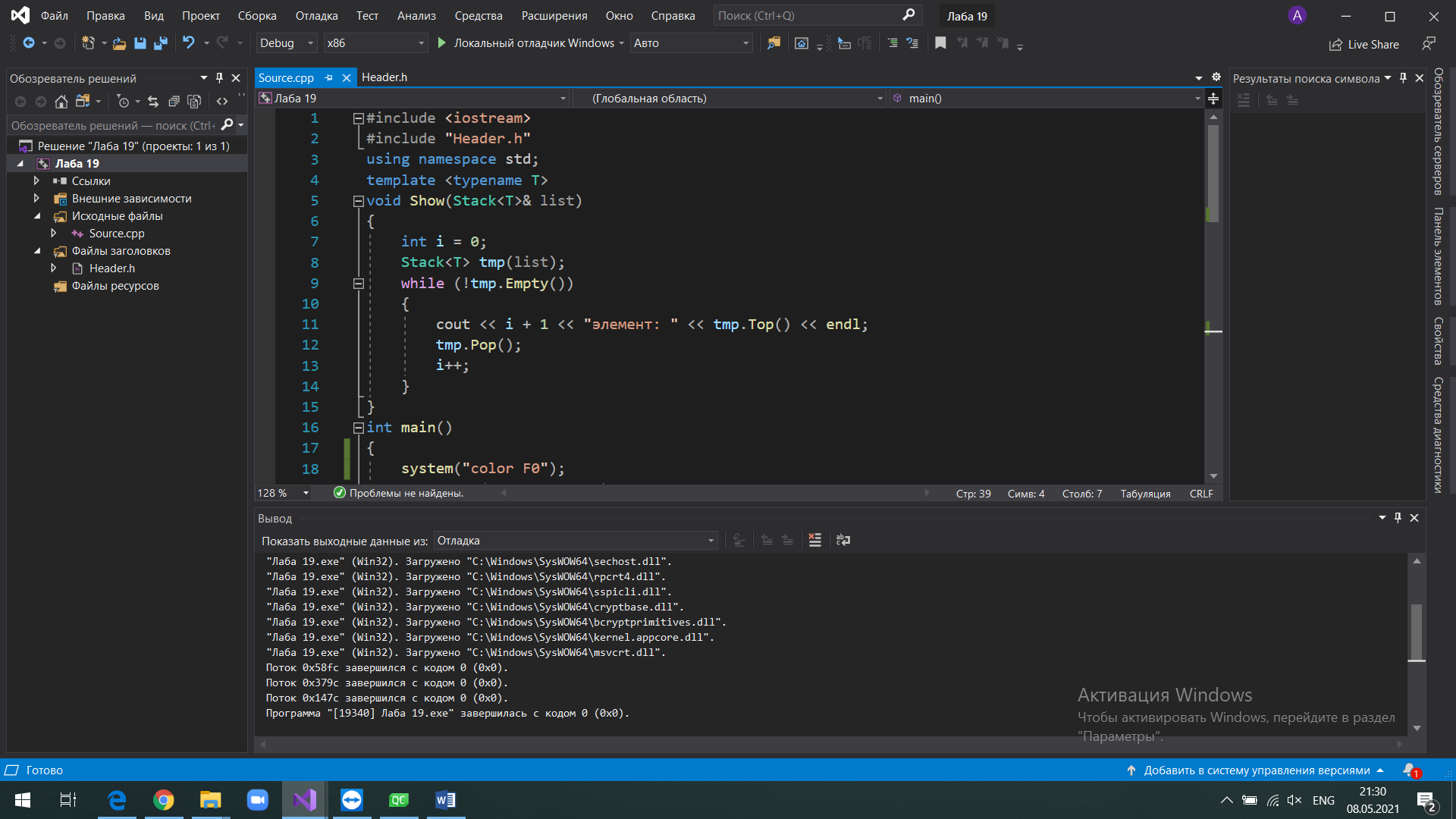
Поле first хранит ссылку на первый элемент стека.

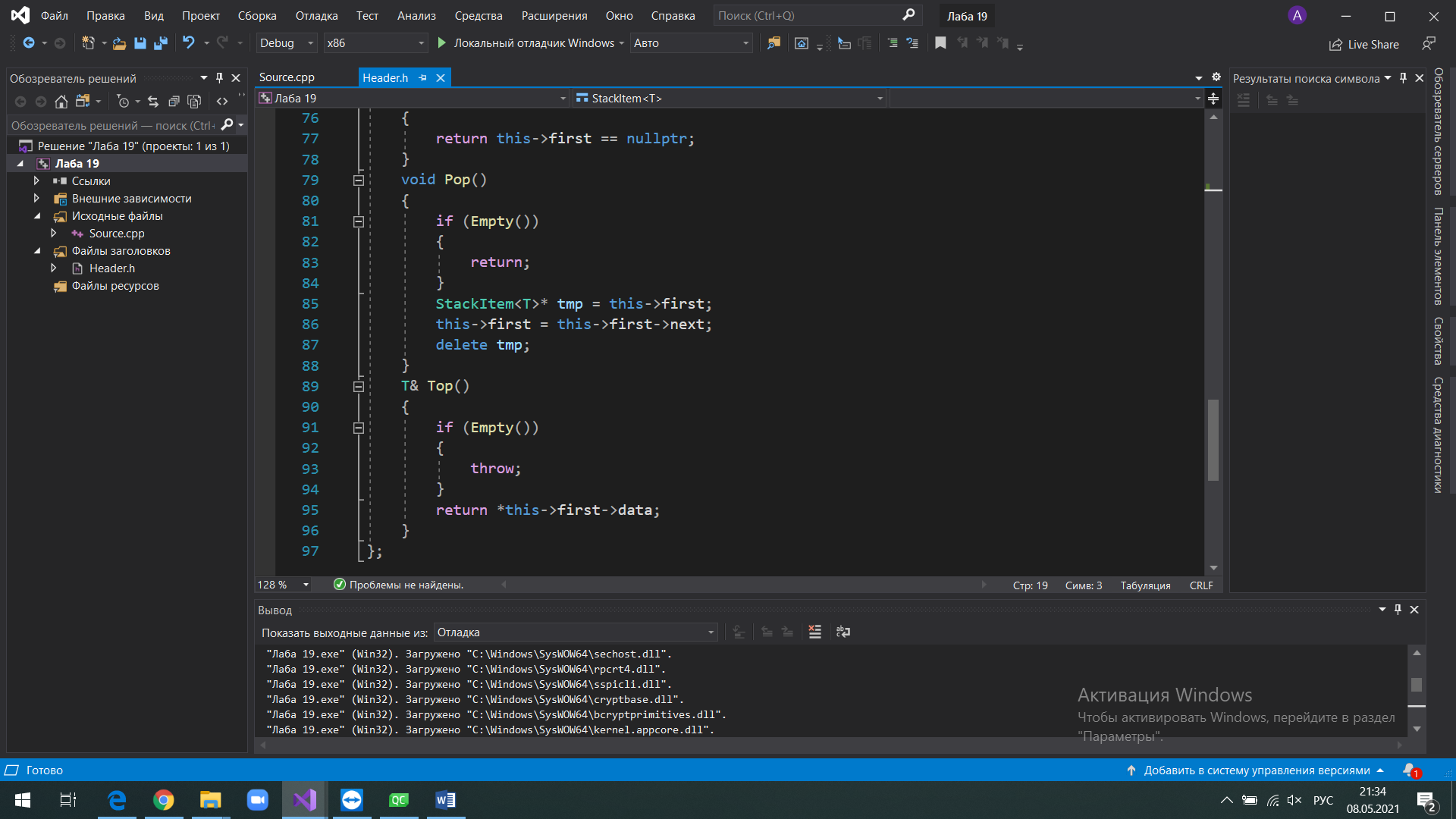
1. Какими операторами будет организован ввод и вывод.
   1. Ввод будет организован с помощью конструктора класса и метода Push().



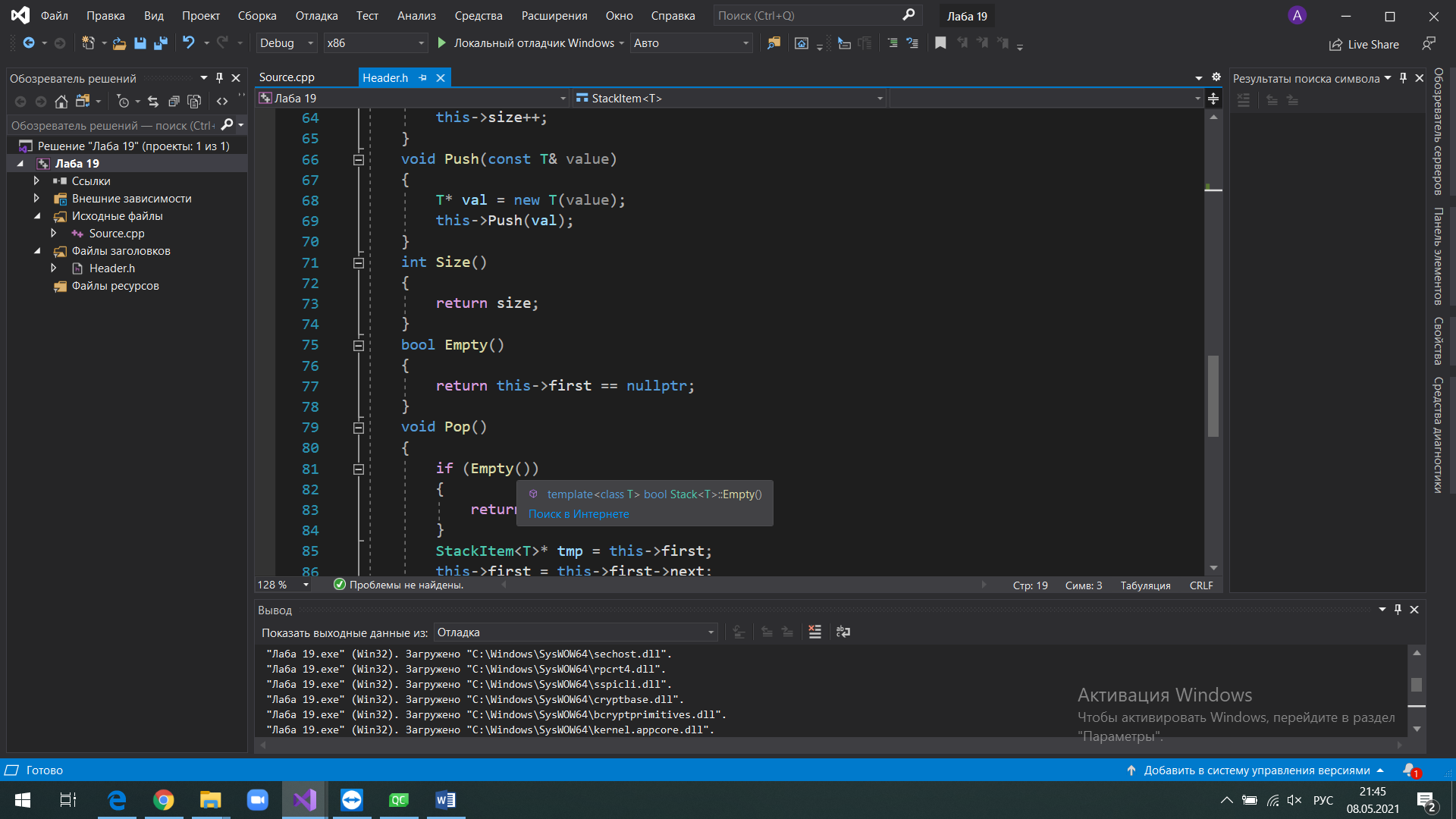


* 1. Вывод элементов организован с помощью глобальной функции Show() и оператора cout. С помощью метода Top() программа получает значение поля data у вершины стека, затем с помощью метода Pop() происходит удаление этого элемента.





1. Какими действиями будут решены поставленные задачи.
   1. Внутри класса организованы методы удаления и добавления элемента, обращения к вершине стека, получения значения размера и также метод, возвращаемый информацию о том пуст ли стек или нет.



* 1. В глобальной функции Show осуществляется обращение к стеку только через вершину, что и требуется при работе со стеком.

1. **Код программы**

Header.h

#pragma once

#include <iostream>

#include <vector>

using namespace std;

template <typename T>

struct StackItem

{

T\* data;

StackItem<T>\* next;

StackItem(T\* Data, StackItem<T>\* Next)

{

this->data = Data;

this->next = Next;

}

~StackItem() {

delete this->data;

}

};

template <typename T>

class Stack

{

int size;

StackItem<T>\* first;

public:

Stack(int size)

{

this->first = 0;

this->size = 0;

for (int i = 0; i < size; i++) {

T\* data = new T;

cout << "Введите " << i + 1 << " элемент: " << endl;

cin >> (\*data);

this->Push(data);

}

}

Stack(Stack<T>& list)

{

this->first = 0;

this->size = 0;

StackItem<T>\* item = list.first;

while (item) {

T\* data = new T(\*item->data);

this->Push(data);

item = item->next;

}

}

~Stack()

{

StackItem<T>\* current = this->first;

while (current) {

StackItem<T>\* rem = current;

current = current->next;

delete rem;

}

}

void Push(T\* value)

{

StackItem<T>\* item = new StackItem<T>(value, this->first);

this->first = item;

this->size++;

}

void Push(const T& value)

{

T\* val = new T(value);

this->Push(val);

}

int Size()

{

return size;

}

bool Empty()

{

return this->first == nullptr;

}

void Pop()

{

if (Empty())

{

return;

}

StackItem<T>\* tmp = this->first;

this->first = this->first->next;

delete tmp;

}

T& Top()

{

if (Empty())

{

throw;

}

return \*this->first->data;

}

};

Main.cpp

#include <iostream>

#include "Header.h"

using namespace std;

template <typename T>

void Show(Stack<T>& list)

{

int i = 0;

Stack<T> tmp(list);

while (!tmp.Empty())

{

cout << i + 1 << "элемент: " << tmp.Top() << endl;

tmp.Pop();

i++;

}

}

int main()

{

system("color F0");

setlocale(LC\_ALL, "rus");

Stack<int> list(5);

cout << "Size:" << list.Size() << endl;

list.Push(new int(2));

list.Push(6);

Stack<int> list2(list);

Stack<int> list3(4);

cout << "List1: " << endl;

Show(list);

cout << "List2: " << endl;

Show(list2);

cout << "List3: " << endl;

Show(list3);

list.Pop();

list.Pop();

list.Pop();

list.Pop();

list.Pop();

list.Pop();

list.Pop();

if (!list.Empty())

cout << list.Top();

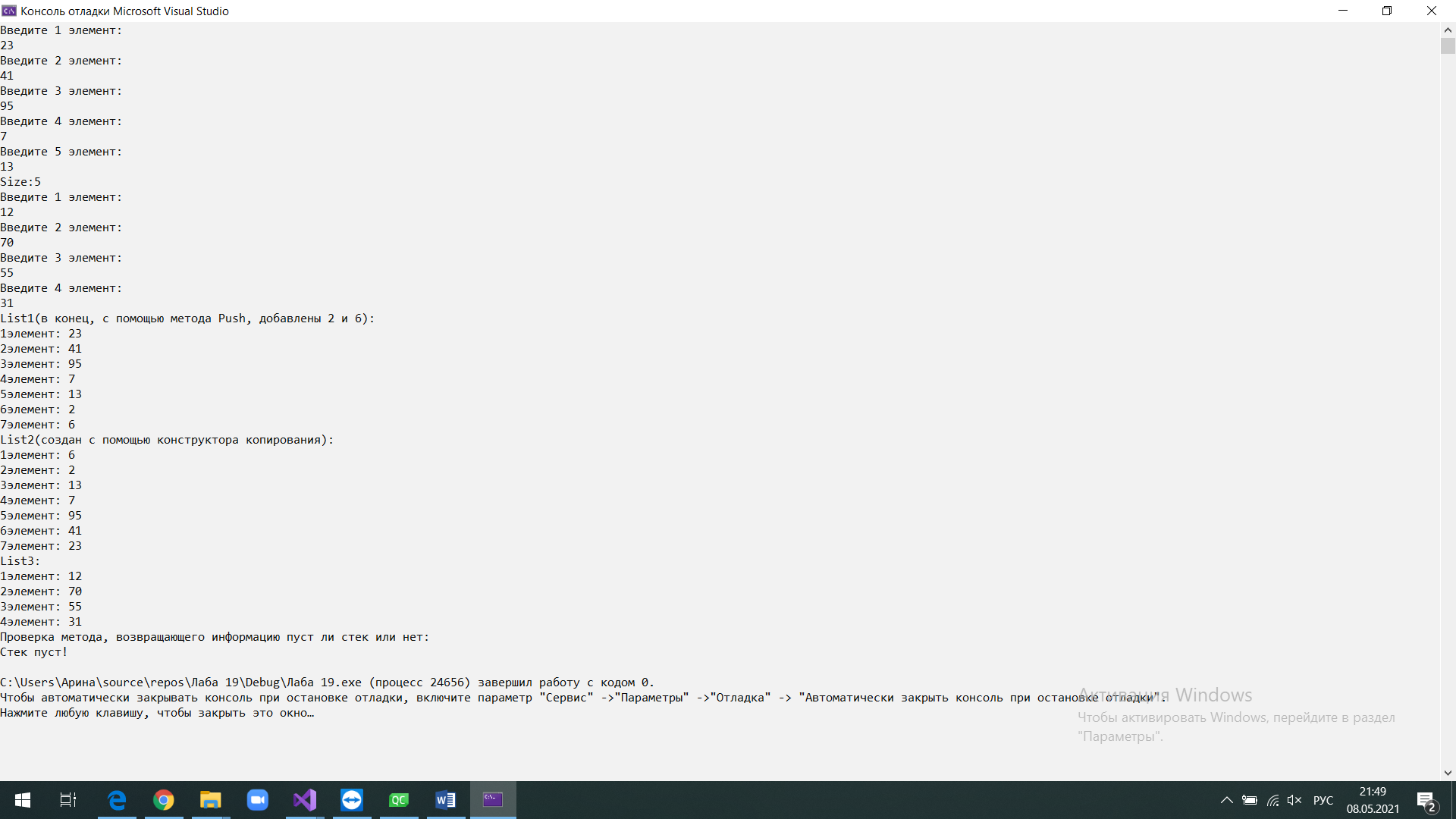
else

cout << "Стек пуст!" << endl;

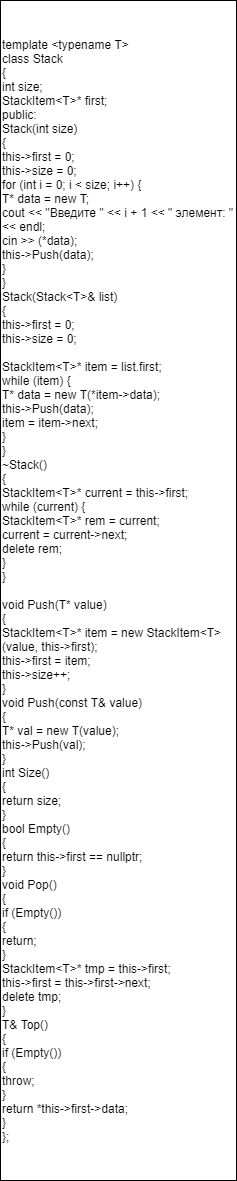
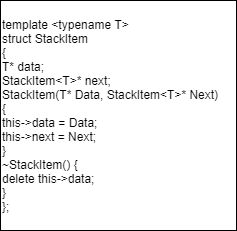
return 0;

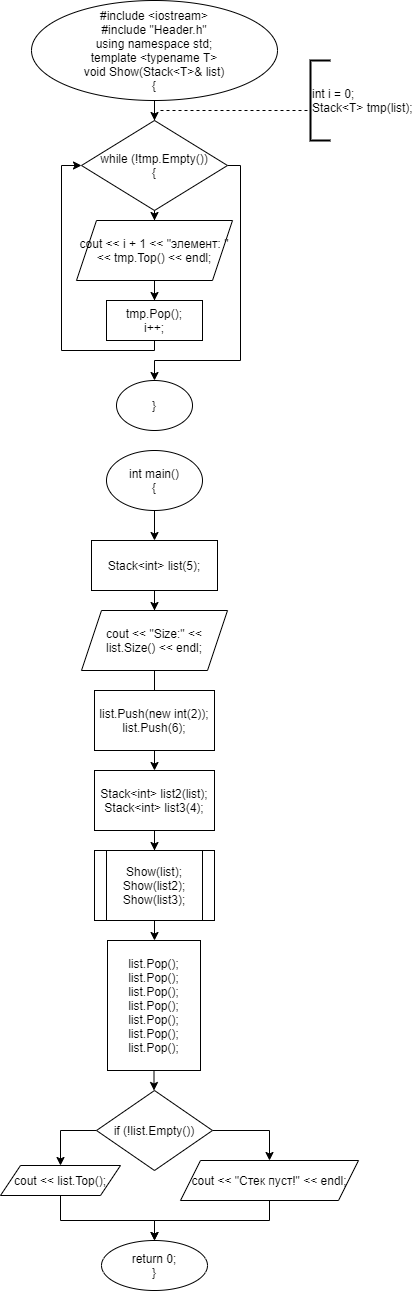
}

1. **Выполнение**



1. **Блок-схема**

****

****